

# 《大数据分析与应用》期末课程项目

## 唐宋古诗数据集处理及诗人风格分析

作 者：喻 云 飞

学 号：3220180897

学 院：计算机学院

## 摘要

我国诗歌文化源远流长，特别是唐宋时期最为丰盛。因此，此次大数据结课项目将针对我国唐宋时期的古诗集进行一系列操作和分析。首先，此次课设使用的数据来自 GitHub，包括 25 万多首宋诗及 5 万多首唐诗，进行重新编号、编码，使用第三方库 ZHConverter 进行繁简体转换，去除无关信息，提取高频词，最终得到根据诗人姓名命名且经过处理的文件。然后，根据每个诗人传作的诗歌使用 jieba 分词进行分词操作，去掉停用词，使用 wordcloud 生成词云进行展示。最后，还尝试使用 tensorflow 和 RNN 来根据每个诗人的风格进行诗歌的自动生成，经过各种调参后，发现训练的模型始终不太理想，遂调研了其他案例，总结了存在的原因。

关键词: 古体诗；中文分词；词云分析；RNN

# 目录

摘要 .....	2
1. 引言 .....	4
2. 数据集操作 .....	5
2.1 数据集的选择.....	5
2.2 数据集操作.....	5
2.3 ZHConverter 介绍.....	5
2.4 关键代码 .....	6
3. 中文分词集词云展示 .....	8
3.1 jieba 分词.....	8
3.2 wordcloud 库.....	8
3.3 代码展示 .....	10
4. 古诗自动生成尝试 .....	11
参考文献 .....	13

# 1. 引言

诗文化中国文化博大精深，特别是古诗文化深刻、生动地体现着中国文化的基本精神。诗文学是语言的艺术，是民族的精神与心灵史，也是文化的主要形态之一。中国文学历经 3000 多年不曾中断，是世界上历史最悠久的文学之一，是中国文化中最重要、最璀璨的部分。另一方面，随着互联网、大数据的不断发展，利用互联网来搜集、分析古诗集将比传统的人工方式更加全面直观。因此此次大数据结课项目将针对我国唐宋时期的古诗集进行一系列操作和分析。

此次课程设计我主要做了以下工作，此片论文也将以此顺序来进行介绍。

第一部分是数据的获取及各种处理。调研了 3 种古诗数据集，最终选择了 github 中的 Chinese-poetry 下的数据集，包括 25 万多首宋诗及 5 万多首唐诗。由于此数据集与我需要的还有些差距，使用 eclipse 编写 TextUtil 项目来对数据集进行重新编号、编码，使用第三方库 ZHConverter 进行繁简体转换，去除无关信息，提取高频词，最终得到根据诗人姓名命名且经过处理的文件。

第二部分主要完成的工作是中文分词和词云展示。根据每个诗人传作的诗歌使用 jieba 分词进行分词操作，去掉停用词，使用 wordcloud 生成词云进行展示。

第三部分尝试使用 tensorflow 和 RNN 来根据每个诗人的风格进行诗歌的自动生成，调研了其他案例，将前面得到的数据集投入 RNN，经过各种调参后，发现训练的模型不太理想，总结了存在的原因。

## 2. 数据集操作

### 2.1 数据集的选择

考察了极速数据 api, chinese-poetry, AncientChinesePoemsDB 三种数据集。

其中通过 api 的方式的优点是不用将数据集全部下载到本地, 内容比较详细, 包含了题目、内容、解释、类型等信息, 但缺点是每次操作都要联网, 并且每天只能访问 100 次。

AncientChinesePoemsDB 是 github 上一个项目, 此数据库中的数据是从郑州大学图书馆网站上爬取下来的, 因为其全唐诗库收录了唐代诗人二千五百二十九人的诗作四万二千八百六三首, 共计九百卷。

chinese-poetry 也是 github 上一个项目, 该诗词数据库包含 5.5 万首唐诗、26 万首宋诗和 2.1 万首宋词。唐宋两朝近 1.4 万古诗人, 和两宋时期 1500 词人, 使用 json 格式, 包含了作者、内容、题目等信息。本课设最终采用该数据集。

### 2.2 数据集操作

为了数据统计方便, 将部分文件进行了重新命名和编号。其中, 编号“0”到“254000”为宋诗, 编号“255000”到“312000”为唐诗。全部为 json 格式。

使用 eclipse 编写 TextUtil 项目来进行操作。该项目包含两个 package。第一个位 entity 实体类, 封装了“Poem”对象, 包含“author”、“paragraphs”、“strains”、“title”4 个属性, 分别是作者、段落、平仄约束和作者。

另一个 package 是 Util 工具类。包含 Constant、Pretreatment、TextOperation 三个类。Constant 类用来定义常量, Pretreatment 类用来封装处理方法, 解析 json 格式、TextOperation 类是主类, 用来对每个文件进行处理。

### 2.3 ZHConverter 介绍

ava-zhconverter 是一个简繁体中文互换的 Java 开源类库。

示例代码:

```
// Instantiation will fetch the property file which load the Chinese character mappings
```

```
ZHConverter converter= ZHConverter.getInstance(ZHConverter.SIMPLI
```

FIED);

```
String simplifiedStr = converter.convert("
    "千里陵陽同陝服，鑿門胙土寄親賢。"，    "曙烟已別黃金殿，晚照重登白
    玉筵。"，
    "江上浮光宜雨後，郡中遠岫列窗前。"，
    "天心待報期年政，留與工師播管弦。"
")将得到该首诗的简体版。
```

## 2.4 关键代码

```
FileInputStream inputStream = new FileInputStream(file);
InputStreamReader inputStreamReader = new InputStreamReader(inputStream, "UTF-8");
BufferedReader in = new BufferedReader(inputStreamReader);
String str;
while ((str = in.readLine()) != null) {
    strbuffer.append(str);
}
```

图 1. 读取单个文件

```
JSONArray array = JSONArray.fromObject(strbuffer.toString());
List<Poem> list = new ArrayList<Poem>();
for (int i = 0; i < array.size(); i++) {
    JSONObject jsonObject = (JSONObject) array.get(i);
    // Poem poem = (Poem) JSONObject.toBean(jsonObject, Poem.class); //
    // 通过JSONObject.toBean()方法进行对象间的转换
    Poem poem = new Poem();
    ZHConverter converter = ZHConverter.getInstance(ZHConverter.SIMPLIFIED);
    poem.setAuthor(converter.convert(jsonObject.getString("author")));
    poem.setParagraphs(converter.convert(jsonObject.getString("paragraphs")));
    list.add(poem);
}
```

图 2. 解析 json 格式，繁体化简体，并添加到 list<Poem>

```
for (int i = 0; i < list.size(); i++) {
    Poem poem = list.get(i);
    FileWriter writer = null;
    // 检查文件夹是否存在，不存在则创建文件夹
    String FilePath = "../dataset_Pretreatment";
    File dir = new File(FilePath);
    if (!dir.exists()) {
        dir.mkdirs();
    }
    // 检查文件是否存在，不存在则创建文件
    File checkFile = new File(dir + "\\total.txt");
    if (!checkFile.exists()) {
        checkFile.createNewFile();
    }
    writer = new FileWriter(checkFile, true);
    writer.append(poem.getParagraphs());
    writer.flush();
    writer.close();
}
```

图 3. 根据实体类列表生成文件

```

public static void main(String[] args) throws IOException {
    for (int i = 0; i <= Constant.MAXLENGTH; i = i + 1000) {
        String FileName = Constant.BasePath + String.valueOf(i) + ".json";
        if (!Pretreatment.EditDataByAuthor(FileName)) {
            System.out.println(FileName + "号数据集处理失败！终止程序!\n");
            break;
        }
        System.out.println(i + "号数据集处理成功！");
    }
}

```

图 4. 循环处理文件

## 3. 中文分词集词云展示

### 3.1 jieba 分词

jieba 分词算法使用了基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能生成词情况所构成的有向无环图(DAG)，再采用了动态规划查找最大概率路径，找出基于词频的最大切分组合，对于未登录词，采用了基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法。

jieba 分词支持三种分词模式：

1. 精确模式, 试图将句子最精确地切开，适合文本分析；
2. 全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义；
3. 搜索引擎模式，在精确模式的基础上，对长词再词切分，提高召回率，适合用于搜索引擎分词。

jieba 分词

1. jieba.cut: 该方法接受三个输入参数：需要分词的字符串；cut\_all 参数用来控制是否采用全模式；HMM 参数用来控制是否适用 HMM 模型
2. jieba.cut\_for\_search: 该方法接受两个参数：需要分词的字符串；是否使用 HMM 模型，该方法适用于搜索引擎构建倒排索引的分词，粒度比较细。
3. 待分词的字符串可以是 unicode 或者 UTF-8 字符串，GBK 字符串。注意不建议直接输入 GBK 字符串，可能无法预料的误解码成 UTF-8，
4. jieba.cut 以及 jieba.cut\_for\_search 返回的结构都是可以得到的 generator(生成器)，可以使用 for 循环来获取分词后得到的每一个词语或者使用
5. jieba.lcut 以及 jieba.lcut\_for\_search 直接返回 list
6. jieba.Tokenizer(dictionary=DEFAULT\_DICT) 新建自定义分词器，可用于同时使用不同字典，jieba.dt 为默认分词器，所有全局分词相关函数都是该分词器的映射。

### 3.2 wordcloud 库

wordcloud 库把词云当作一个 WordCloud 对象。wordcloud.WordCloud() 代表一个文本对应的词云。可以根据文本中词语出现的频率等参数绘制词云。常用函数如下：

- font\_path : string //字体路径，需要展现什么字体就把该字体路径+



后缀名写上，如：font\_path = '黑体.ttf'

- width : int (default=400) //输出的画布宽度，默认为 400 像素
- height : int (default=200) //输出的画布高度，默认为 200 像素
- prefer\_horizontal : float (default=0.90) //词语水平方向排版出现的频率，默认 0.9 （所以词语垂直方向排版出现频率为 0.1 ）
- mask : nd-array or None (default=None) //如果参数为空，则使用二维遮罩绘制词云。如果 mask 非空，设置的宽高值将被忽略，遮罩形状被 mask 取代。
- 除全白（#FFFFFF）的部分将不会绘制，其余部分会用于绘制词云。如：  
bg\_pic = imread('读取一张图片.png')，
- 背景图片的画布一定要设置为白色（#FFFFFF），然后显示的形状为不是白色的其他颜色。可以用 ps 工具将自己要显示的形状复制到一个纯白色的画布上再保存，就 ok 了。
- scale : float (default=1) //按照比例进行放大画布，如设置为 1.5，则长和宽都是原来画布的 1.5 倍。
- min\_font\_size : int (default=4) //显示的最小的字体大小
- font\_step : int (default=1) //字体步长，如果步长大于 1，会加快运算但是可能导致结果出现较大的误差。
- max\_words : number (default=200) //要显示的词的最大个数
- stopwords : set of strings or None //设置需要屏蔽的词，如果为空，则使用内置的 STOPWORDS
- background\_color : color value (default=" black" ) //背景颜色，如 background\_color='white'，背景颜色为白色。
- max\_font\_size : int or None (default=None) //显示的最大的字体大小
- mode : string (default=" RGB" ) // 当 参 数 为 “RGBA” 并且 background\_color 不为空时，背景为透明。
- relative\_scaling : float (default=.5) //词频和字体大小的关联性
- color\_func : callable, default=None //生成新颜色的函数，如果为空，则使用 self.color\_func
- regexp : string or None (optional) //使用正则表达式分隔输入的文本
- collocations : bool, default=True //是否包括两个词的搭配
- colormap : string or matplotlib colormap, default=" viridis"

//给每个单词随机分配颜色，若指定 color\_func，则忽略该方法。

- fit\_words(frequencies) //根据词频生成词云【frequencies，为字典类型】
- generate(text) //根据文本生成词云
- generate\_from\_frequencies(frequencies[, ...]) //根据词频生成词云
- generate\_from\_text(text) //根据文本生成词云
- process\_text(text) //将长文本分词并去除屏蔽词（此处指英语，中文分词还是需要自己用别的库先行实现，使用上面的fit\_words(frequencies) ）。
- recolor([random\_state, color\_func, colormap]) //对现有输出重新着色。重新上色会比重新生成整个词云快很多。
- to\_array() //转化为 numpy array
- to\_file(filename) //输出到文件

### 3.3 代码展示

```
import os
import jieba
import matplotlib.pyplot as plt
from wordcloud import WordCloud

path = "F:\\JeeWorkspace\\dataset_Pretreatment1\\"
files = os.listdir(path)

for file in files:
    txt_path = path + file

    # 读取txt文件
    content = open(txt_path, 'r', errors='ignore').read()
    # 使用jieba进行分词
    jieba_cut = " ".join(jieba.cut(content, cut_all=False))
    # 字体
    font = "C:\\Users\\YYF\\Desktop\\Poem\\Font\\WeiRuanZhengHei.ttf"
    # 停用词
    stopwords = {}.fromkeys([line.rstrip() for line in open('C:\\Users\\YYF\\Desktop\\Poem\\StopWord.txt')])

    final = ''
    for seg in jieba_cut:
        if seg not in stopwords:
            final = final + seg

    jieba_cut2 = " ".join(jieba.cut(final, cut_all=False))

    # 词云准备
    cloud = WordCloud(
        max_words=100,
        background_color="white",
        #mask=background,
        font_path=font,
        width=600,
        height=460,
        margin=2)

    # 生成词云
    word_cloud = cloud.generate(jieba_cut)

    # 存储词云
    cloud.to_file(os.getcwd()+"\\wordcloud\\" + file.replace("txt", "png"))

print("生成完成!")
```

图 5 分词、去停用词、词云生成与存储代码展示

## 4. 古诗自动生成尝试

循环网络(Recurrent Neural Network)旨在对序列进行建模,允许用户在保留结构信息的同时处理序列。由于语言的顺序性,它们在 NLP 任务中特别有用。循环网络由循环组成,其中特定层的输出作为输入传递回同一层。这允许信息持久化并捕获长期依赖性,例如序列中出现的依赖性。RNN 最受欢迎的实现之一是长短期记忆(LSTM),它是为了减轻消失的梯度问题而引入的。随着序列变长,当前单词与依赖上下文之间的距离变长。然而,这意味着序列后面步骤中的误差梯度在反向传播中迅速减小,并且没有达到较早的输入信号,因此梯度“消失”。这使得捕获相关信息变得非常困难。LSTM 引入了一个充当存储单元的向量,可以随时保留渐变。通过门控可以被认为是逻辑门的组件来控制对存储器单元的访问。

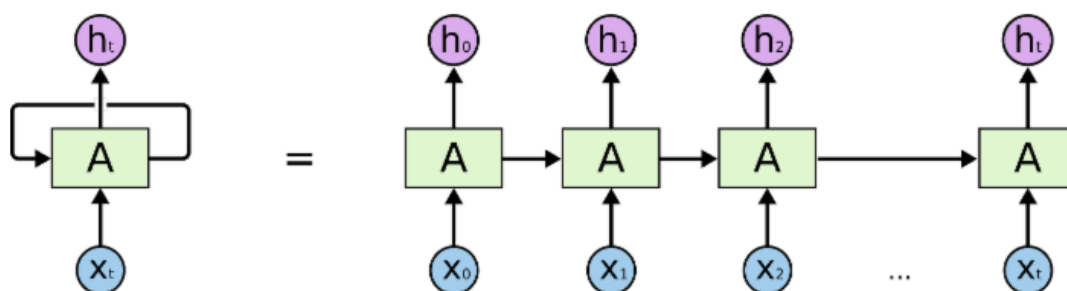


图 6 展开的 RNN 神经网络

使用 TensorFlow 中的 char-rnn 代码,将前面处理好的文本投入 CNN 中进行训练和生成,得到以下结果:

```
Tue Jan 15 14:18:50 2019 i: 19 step: 2549 batch_loss: 5.1939616
Tue Jan 15 14:18:50 2019 i: 19 step: 2550 batch_loss: 5.148732
Tue Jan 15 14:18:50 2019 i: 19 step: 2551 batch_loss: 5.203025
Tue Jan 15 14:18:51 2019 i: 19 step: 2552 batch_loss: 5.0888157
Tue Jan 15 14:18:51 2019 i: 19 step: 2553 batch_loss: 5.0993657
Tue Jan 15 14:18:51 2019 i: 19 step: 2554 batch_loss: 5.232712
Tue Jan 15 14:18:51 2019 i: 19 step: 2555 batch_loss: 5.232076
Tue Jan 15 14:18:52 2019 i: 19 step: 2556 batch_loss: 5.1736627
Tue Jan 15 14:18:52 2019 i: 19 step: 2557 batch_loss: 4.9924808
Tue Jan 15 14:18:52 2019 i: 19 step: 2558 batch_loss: 5.0772753
Tue Jan 15 14:18:52 2019 i: 19 step: 2559 batch_loss: 5.293551
```

图 6. epoch=20 的训练结果

```
袁民曾隔端旆光香开风。
第山莫浪叙,里枝若别伤。
鸟驰宸榆发,庭尔地洒错。
笛而满雪契,迎灰出乐流。
至思国梅尽,天醉驷如时。
宁长逢重脍,鸣怀白期今。
不廓三乡林,浩出未波时。
今云楚年跃,光望如岁
```

图 7. 生成结果

可以看出,有些格式还是存在问题,上网看了看了一下使用同样模型的结果,

发现他们的训练次数都是几万左右，再观测他们的数据也基本是使用全唐时期的古诗（5w 多首）作为训练数据来进行训练。于是，推测训练效果不佳的原始是训练数据太小了，将 30 万首诗按照 1 万个诗人进行划分，写诗最多的陆游的文件大小也才 1.5M，因此，数据太少，模型不佳。

## 5. 总结

通过此次课程设计，使我有如下收获。一是将课堂所学到分词，词云，CNN等理论知识结合代码运行了起来，有了直观的感受。二是学会 jieba 库、wordcloud 库、ZHConverter 库等第三方库的简单实用。三是提高了 JAVA，Python 的编程技巧。四是完成了模型的训练，虽然使用了开源代码，但对于训练过程也有了自己的感受。

另一方面，此次课设也获得了以下成果。一是对于原来分布散乱的数据集进行了统一的根据作者姓名分类的划分，二是对于每位唐宋时期的诗人，都能根据他创作的诗来生成属于他自己的词云，分辨他人对诗人有更加直观的感受。三是对于“根据诗人来生成属于它自身风格的诗”这一想法进行了尝试，对于后面想进行这方面研究的同学提供了参考。

## 参考文献

[1]唐家渝, 孙茂松. 新媒体中的词云: 内容简明表达的一种可视化形式[J]. 中国传媒科技, 2013(11): 18-19.

[2] 梁喜涛, 顾磊. 中文分词与词性标注研究[J]. 计算机技术与发展, 2015, 25(02): 175-180.

[3]刘铮扬. 基于深度学习的中国诗歌自动生成算法研究[D]. 湖南师范大学, 2018.

[4]冯思齐, 吕钢. 利用 RNN 模型生成中国古典诗歌[J]. 通讯世界, 2018(02): 361-363.